Combining Logical Abduction and Statistical Induction: Discovering Written Primitives with Human Knowledge*

Wang-Zhou Dai and **Zhi-Hua Zhou**
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210023, China
{daiwz, zhouzh}@lamda.nju.edu.cn

Abstract

In many real tasks there are human knowledge expressed in logic formulae as well as data samples described by raw features (e.g., pixels, strings). It is popular to apply SRL or PILP techniques to exploit human knowledge through learning of symbolic data, or statistical learning techniques to learn from the raw data samples; however, it is often desired to directly exploit these logic formulae on raw data processing, like human beings utilizing knowledge to guide perception. In this paper, we propose an approach, *LASIN*, which combines Logical Abduction and Statistical INduction. The *LASIN* approach generates candidate hypotheses based on the abduction of first-order formulae, and then, the hypotheses are exploited as constraints for statistical induction. We apply the *LASIN* approach to the learning of representation of written primitives, where a primitive is a basic component in human writing. Our results show that the discovered primitives are reasonable for human perception, and these primitives, if used in learning tasks such as classification and domain adaptation, lead to better performances than simply applying feature learning based on raw data only.

1 Introduction

Unifying statistical and symbolic machine learning has been mentioned more and more frequently (Russell 2015; Tenenbaum et al. 2011). In fact, almost 30 years ago, Donald Michie (1988) had already pointed out that the development of machine learning should be able to manipulate symbolic representation. Many approaches have been proposed to fulfill this goal, which can be roughly categorized into two types: Statistical Relational Learning (SRL) (Getoor and Taskar 2007) and Probabilistic Inductive Logic Programming (PILP) (De Raedt and Kersting 2008). Owing to the ability to combine the syntactic and semantic expressiveness of first-order logic with the compositional semantics of probabilistic model (Koller and Friedman 2009), these approaches have achieved great success in many areas including natural language processing (Wang, Mazaitis, and Cohen 2013), robotics (Nitti, De Laet, and De Raedt 2014), bioinformatics (De Maeyer et al. 2013), etc.

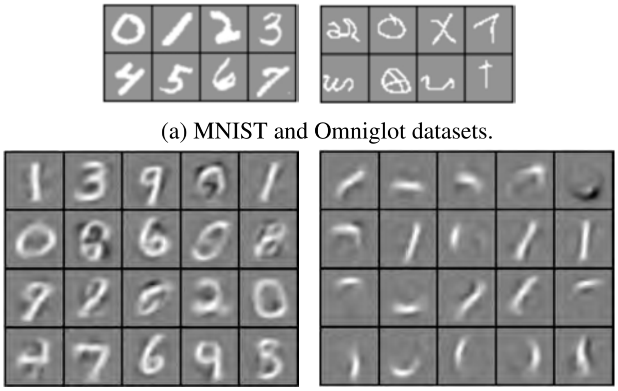
Most approaches for unifying statistical and symbolic learning were designed for applications in symbolic domains (Russell 2015). In many real tasks, however, human knowledge expressed in first-order logic (FOL) and data samples described by raw features exist simultaneously. For example, if we want to discover written primitives from images of handwritten characters, human knowledge about pen strokes is worth being exploited (see Figure 1). On one hand, this kind of background knowledge, which can be easily expressed with symbolic language like FOL, are difficult to inject into common statistical learning (Getoor and Taskar 2007). On the other hand, typical approaches for unifying statistics and logic are capable of exploiting the FOL expressed human knowledge, yet they are seldom designed for raw data inputs (Russell 2015).

In this paper, we define the *concept extraction* problem. Similar to traditional representation learning (Bengio, Courville, and Vincent 2013), the goal of *concept extraction* is to learn a set of features from the raw input space. The difference lies in the fact is that *concept extraction* is facilitated with an FOL knowledge base.

To address *concept extraction*, we draw inspiration from human cognition. Perception, a human cognitive process in charging of the organization, identification and interpretation of raw sensory information (Schacter, Gilbert, and Wegner 2011), faces this kind of problem at almost every second in our life. According to Charles S. Peirce (1955), human perception is a kind of abduction, i.e., a form of logical inference going from an observation to a theory which may account for the observation. As psychologists admitted, Peirce's theory provides "a midway between a seeing and a thinking" (Tiercelin 2005; Magnani 2009).

Inspired by Peirce's abductive perception theory, we propose the *LASIN* approach that combines Logical Abduction and Statistical INduction for *concept extraction*. Firstly, we exploit abductive logic theory (Kakas, Kowalski, and Toni 1992) to generate ground hypotheses from the raw inputs. These hypotheses are then exploited as constraints for statistical induction to obtain a concept dictionary. Finally, the concepts are tested on the raw data and get accepted or revised for improvements. Experimental results show that the proposed approach is able to exploit FOL knowledge base and extract written primitives that are reasonable for human perception, where a primitive is a basic component in hu-

*This research was supported by the 973 Program (2014CB 340501) and NSFC (61333014).



(a) MNIST and Omniglot datasets.

(b) Results of Sparse Coding. (c) Results of *LASIN*.

Figure 1: (a): Examples of handwritten character datasets. (b) and (c): Dictionaries extracted from MNIST without/with knowledge of strokes.

man writing. Furthermore, the primitives extracted with appropriate background knowledge can be beneficial to machine learning tasks such as classification and domain adaptation. Although the experiments are conducted on handwritten data, the *LASIN* approach is applicable to general problems as long as the human knowledge and raw data can be bridged by FOL formulae.

The rest of this paper is organized as follows: We first introduce the related works and formal definition of the *concept learning* task; then the details of *LASIN* algorithm are described; finally we report the empirical evaluations with discussions and then conclude.

2 Related Works

Several SRL/PILP approaches have been proposed to analyze images or handwritten data (Antanas et al. 2014; Shivram et al. 2014). These approaches could not directly handle the data represented by raw features. They usually perform feature extractions on the raw data at first, and then use statistical models to discover the symbols before relational learning. There are also approaches using statistical learning after logic learning (Dai and Zhou 2015), which could not be directly applied on raw data either.

Representation learning is a class of approaches that learn representations of the data, aiming at the extraction of useful information when building classifiers or other predictors (Bengio, Courville, and Vincent 2013). These approaches have achieved great success in practice. Many of them have been successfully applied to handwritten character recognition tasks, such as sparse coding (Lee et al. 2007; Olshausen and Field 1996) , manifold learning (Yu, Zhang, and Gong 2009), deep neural networks (Cireşan et al. 2010). Most of representation learning techniques are based on statistical optimization that is subsymbolic and can hardly introduce symbolic knowledge like humans (Fiser et al. 2010).

Recent progress in artificial intelligence (AI) has renewed interest in building systems that learn and think like people (Lake et al. 2016). A representative work in this branch similar to this paper is (Lake, Salakhutdinov, and Tenenbaum 2015), which learns the concept of strokes through induction of Bayesian programs. The difference lies in the fact that their inputs and background knowledge are not raw pixels and FOL but pen trajectories.

This paper focuses on exploiting FOL-represented human knowledge in raw data processing like abductive perception. We apply the proposed *LASIN* on handwritten data in this paper, since Douglas Hofstadter had suggested that "the problem of recognizing characters in all the ways people do contains most if not all of the fundamental challenges of AI" (Hofstadter 1985).

Recently the concept "learnware" is proposed (Zhou 2016), which is a pre-trained reusable model facilitated by specifications to be matched with user requests. The user requests deliver task requirement, actually a kind of knowledge, whereas logic description is an important option of composing the specifications/requests. Thus, studying the usage of logic formulae in statistical learning will also help explore some learnware implementation possibilities.

3 Problem Setting

In this section we formally present the task of *concept extraction*. Intuitively, *concept extraction* constrains the extracted features to be coherent with a well-defined background knowledge base. In this paper, we choose first-order logic (FOL) as the language of background knowledge.

A first-order alphabet is composed of constants, variables, functions, predicates, quantifiers and connectives. Constants represent objects in domain, e.g., "1" and "$anna$". Variables range over the constants, e.g., "X", and "$Person$". Functions represent mappings from tuples of objects to objects, e.g., "$s(X)$" can be used to represent $X + 1$. Predicates represent relations among objects or attributes, e.g., "$friends(X, Y)$" means X and Y are friends. Quantifiers "$\forall$" and "$\exists$" constrain the range of variables. "$\forall X(p(X))$" and "$\exists X(p(X))$" are identical to say "$p(X)$" is true for all X and some X, respectively. Connectives are "$\leftarrow$" for implication, "$\wedge$" for conjunction, "$\vee$" for disjunction, "$\neg$" for negation and "$=$" for equality. A term is a constant, a variable or a function symbol immediately followed by a bracketed n-tuple of terms, e.g., "bob", "X" and "$s(s(s(0)))$". An atom is a predicate symbol applied to a tuple of terms, e.g., "$greater\_than(X, 2)$". A term or atom is said to be *ground* if and only if it does not contain any variable. Formulae are inductively defined by the following rules: 1) if P is a predicate symbol and $T_1, \ldots, T_n$ are terms then $p(T_1, \ldots, T_n)$ is a formula; 2) if ϕ is a formula, then $\neg\phi$ is a formula; 3) if ϕ and ψ are formulae, then $\phi \leftarrow \psi$ is a formula, and similar rules apply to other binary logical connectives; 4) if ϕ is a formula and X is a variable, then $\forall X(\phi)$ and $\exists X(\phi)$ are formulae. For example, $\forall X(natural(X) \leftarrow integer(X) \wedge X \geq 0)$ is a first-order logical formula defining natural numbers. A background knowledge base KB is a set of first-order formulae, and a formula t satisfying KB is denoted as $KB \vDash t$, e.g., if KB is the previous definition of natural number, we have $KB \vDash natural(1)$ and $KB \nvDash natural(-9)$.

Concept extraction is formally defined as follows. The input consists of a set of training instances $\mathbf{x} = \{x^{(1)}, \ldots, x^{(m)}\}$ with a background knowledge base KB, where $x^{(i)} \in \mathbb{R}^n$; KB is a set of first-order logic formulae. The task is to extract a dictionary $D = \{b_1, \ldots, b_s\}$ from $\mathbf{x}$ such that $\mathbf{x}$ can be accurately reconstructed by D, where each basis vector $b_j \in \mathbb{R}^n$ is a concept corresponding to KB, i.e., $\exists$ predicate $p \in KB$ such that $KB \vDash p(b_j)$.

Concept extraction can be seen as an analogue to human perception, for example:

Example 1 *Suppose we are seeing images of handwritten Arabic numbers. Typical background knowledge we hold is that characters are written stroke by stroke. We also know that strokes are continuous ink trajectories. In order to be written smoothly, every sub-stroke should not have drastic direction change. With this knowledge, we can easily accomplish two tasks: i) discover commonly appeared written primitives and use their spacial relations to code the characters; More importantly, ii) the discovered primitives can be of help for learning other handwritten characters.*

Background knowledge in this example can be conveniently expressed by an FOL knowledge base:

$$\begin{aligned}
&\mathbf{KB1}: \\
&\forall S(stroke(S) \leftarrow S = \{P_1, P_2, \ldots\} \\
&\qquad\qquad \wedge\, sub_strk(P_1, P_2, P_3) \\
&\qquad\qquad \wedge\, sub_strk(P_2, P_3, P_4) \wedge \cdots). \qquad (1) \\
&\forall A \forall B \forall C(sub_strk(A, B, C) \leftarrow \\
&\qquad\qquad ink(\mathbf{AB}) \wedge ink(\mathbf{BC}) \\
&\qquad\qquad \wedge\, angle(\mathbf{AB}, \mathbf{BC}) < \alpha).
\end{aligned}$$

where $stroke(S)$ determines whether a point sequence $S = \{P_1, P_2, \ldots\}$ forms a trajectory of pen stroke, and each point $P_i = (X_i, Y_i)$ is a coordinate on image; $sub\_strk(A, B, C)$ is sub-stroke constraint on two sequential ink segments $\mathbf{AB}$ and $\mathbf{BC}$; $ink(\mathbf{AB})$ is true when there exists ink on line segment $\mathbf{AB}$ (with end points A and B); $angle(\mathbf{AB}, \mathbf{BC})$ calculates the angle between the two vectors; α is a positive number (e.g. $\frac{\pi}{2}$) to limit the turning angle from $\mathbf{AB}$ to $\mathbf{BC}$.

The main challenge for *concept extraction* is how to constrain the searching process in raw feature space with KB. Firstly, general FOL constraints are complicated and mostly indifferentiable for statistical optimization. Secondly, raw data samples usually distribute in $\mathbb{R}^n$ which contains an infinite number of possible groundings for symbolic models (Russell 2015).

4 The LASIN Approach

To tackle the *concept extraction* task, we propose the *LASIN* approach. The main idea is to constrain the statistical learning by feeding it with specific input data filtered by KB.

We follow the framework of *active hypothesis-testing* process (Pyszczynski and Greenberg 1987). It is a problem solving process where a human encounters novel or unexpected events, which majorly consists of several sequential phases: 1) selection of a hypothesis for testing; 2) search for information relevant to the hypothesis; 3) assessment of the fit between the pattern of information implicated by the hypothesis and accessed during the information search stage; 4) evaluation of the fit, accept, reject or update the hypothesis. An outline of the proposed algorithm is shown as Algorithm 1.

Algorithm 1: *LASIN*

Input : Training instances $\mathbf{x}$; FOL knowledge base KB.
Output: A dictionary $D = \{d_j\}$ of target concepts.

$D_0 = \Phi$;
while $t <$ *turn limit* **or** $||error|| >$ *threshold* **do**
  `/* test the dictionary learned from previously learned dictionary */`
  $\mathbf{x}' =$ `Reconstruct` $(D_{t-1}, \mathbf{x})$;
  $error = loss(\mathbf{x}', \mathbf{x})$;
  `/* logic abduction */`
  Hypotheses = `Abduce` $(error, KB)$;
  `/* statistical induction */`
  $D_t =$ `SparseCoding` (Hypotheses);
  $t = t + 1$;
end
$D = D_t$;

Logical Abduction

The first step of *LASIN* is to generate ground hypotheses that account for raw data samples in $\mathbf{x}$ by logical abduction.

According to Charles S. Peirce, *Abduction* is a kind of logic inference. Different to *deduction* (from general rules to particular cases) and *induction* (from cases to rules), *abduction* is the inference process of forming a ground hypothesis that explains observed phenomena (Peirce 1955).

Abduction does not only involve in symbolic reasoning. In fact, vision is a good example of human applying abductive reasoning in subsymbolic scenarios (Park 2015; Dai, Muggleton, and Zhou 2015). For example, when we see a picture of a car, the pixels just tell us about its color and shape on one side; however we still can guess about its appearance in unobserved directions. Furthermore, we can even figure out its model type and many other information. Obviously, human can abduce logical symbols (e.g., model types) with just raw visual inputs (e.g., pixels).

Logical abduction has been applied to symbolic machine learning before (Tamaddoni-Nezhad et al. 2006). Here we give a brief introduction to abductive logic theory (Kakas, Kowalski, and Toni 1992):

Definition 1 *Given an abductive logic theory (P, A) where P is a logic program, A is set of abducible predicates in the logic program P. For an observation O, Δ is an abductive explanation consisting of a set of ground abducible atoms on the predicates A such that $P \cup \Delta \vDash O$.*

For knowledge base KB1, we can define an abductive logic theory like this: the observations O are the images of characters $x^{(i)} \in \mathbf{x}$; the set of abducible predicate $A = \{Stroke/1\}$; the abductive program P is an FOL clause simply saying "a character is composed by strokes":

$$\begin{aligned}
&\forall C(character(C) \leftarrow \\
&\qquad C = \{S_1, S_2, \ldots\} \wedge Stroke(S_1) \wedge \ldots).
\end{aligned} \qquad (2)$$

Observing a fact $character(x^{(i)})$, the logical abduction procedure will try to abduce a possible explanation $\Delta^{(i)}$ – in this example, a sequence of "strokes" in $x^{(i)}$. When each time the abduction solver tries to find a ground example of $Stroke(S)$, it will consult its definition in KB1 and finally queries about the most basic facts such as ink line segments and ink points using Logical Vision (Dai, Muggleton, and Zhou 2015). To increase the efficiency, we apply greedy search for hypotheses abduction and use random sampling for basic facts (ink points) discovery. The time complexity of hypothesis abduction on each instance is $O(s \log s)$ as it is implemented with a recursive logic program, where s is the number of sampled ink points on an instance.

The abduced groundings form a conjunction to explain the raw dataset $\mathbf{x}$. It is worth noticing that we should not simply equate abduction with inference to the *best* explanation because the result of abduction is not unique, which implies that there must be other processes between logical abduction and getting the best explanation. Here we suggest to take *statistical induction* as a candidate.

Statistical Induction

After the logical abduction, $\mathbf{h} = \{\Delta\}$ is obtained. It is the set of all abductive explanations from the training instances $\mathbf{x}$. Then, a statistical induction procedure is called to select the "best hypotheses". Finally, the selected hypotheses are tested in the original raw data and get accepted or revised.

In this paper, we use Sparse Coding (SC) (Lee et al. 2007) for statistical induction.[1]. The optimization objective of SC encourages each input to be reconstructed well by a set of sparse codes and the extracted dictionary.

The statistical induction tries to find a small set of "best hypotheses", and the resulted objective function can be written as follows:

$$\begin{aligned} &\min_{b,a} \sum_k ||h^{(k)} - \sum_j a_j^{(k)} b_j||_2^2 + \beta ||a^{(k)}||_1 \\ \text{s.t.} \quad & ||b_j||_2 \leq 1, \forall j \in 1, \ldots, s. \\ \text{where} \quad & \forall h^{(k)} \exists x (x \in \mathbf{x} \wedge h^{(k)} \subset x \wedge KB \vDash p(h^{(k)})). \end{aligned}$$

where $h^{(k)} \subset x$ means $h^{(k)}$ is a ground explanatory hypothesis abduced from x; p is the predicate of target concept in KB; $D = \{b_1, b_2, \ldots, b_s\}$ are the *basis vectors* (dictionary) with each $b_j \in \mathbb{R}^n$ as an extracted concept; $a = \{a^{(1)}, a^{(2)}, \ldots, a^{(n)}\}$ are the *codes*; $a_j^{(k)}$ is the activation of the basis b_j for input $h^{(k)}$. The learned bases are abstractions of $h^{(k)}$ and can be seen as the induced "best explanations". Because $\{h^{(k)}\}$ are obtained from logical abduction, they are guaranteed to satisfy the constraint in this objective function.

The final step of *LASIN* is to test the quality of the abstracted hypotheses D. More precisely, it will use D to reconstruct the original data $\mathbf{x}$ and compute the error of the reconstructed $\mathbf{x}'$. Depending on the quality of $\mathbf{x}'$, the algorithm will choose either to revise the hypotheses $\mathbf{h}$ by doing more abductions based on the difference between $\mathbf{x}$ and $\mathbf{x}'$ or to return current D as output.

[1] Although using Sparse Coding may sacrifice some background knowledge coherency of the learned dictionary (with a low likelihood because we use patch-based SC), it benefits the experiments on unbiasedness and convenience. In order to reduce the bias of *concept extraction* to ensure the learned concepts be general enough for domain adaptation, we do not use feature selection since most of them are supervised. Furthermore, it is natural to design a comparison between unsupervised representation learning with/without FOL background knowledge on raw pixel image inputs, e.g., "abduction/no_abduction+SC". If we use abduction+selection, it will be difficult to design experiments such as "no_abduction+selection" for raw data inputs.

5 Empirical Evaluation

In this section we report two experimental results of *LASIN* on 3 real handwritten characters datasets. Some examples of the datasets and results are illustrated with Figure 1 and 3. We compare *LASIN* with original sparse coding as the baseline since it is a widely used representation learning approach and has been proved to be successful in practice (Lee et al. 2007).

The sparse coding and other clustering models in the experiments are implemented by the mlpack toolbox (Curtin et al. 2013). Logical abduction is implemented by using SWI-Prolog (Wielemaker et al. 2012).

Devanagari Primitives Discovery

We use **HPL-Devanagari** (Bharath and Madhvanath 2010) dataset in this task. This dataset contains approximately 270 samples of each of 111 Devanagari characters written by over 100 native Hindi speakers. Each Devanagari character is constructed by some primitive strokes (Kopparapu and Lajish 2014), shown in Figure 2. Different to OCR task which uses writing trajectories to recognize the characters, in this experiment, we only use the raw images to extract $|D| = 200$ handwritten primitives from the characters. We compare the stroke (Knowledge base KB1) based *LASIN* with original sparse coding trained with same input data and default parameters.

Samples of the results of *LASIN* and sparse coding are shown in Figure 3c and Figure 3d, respectively. Samples of the ground hypotheses abduced by stroke based *LASIN* are shown in Figure 3b. Comparing Figure 2 with Figure 3c and 3d, we can observe that augmented with human knowledge about strokes, *LASIN* can extract written primitives that are more reasonable for human cognition. This is because the result of logical abduction (as in Figure 3b) constrain statistical induction to search for models in a local area that close to human perceived concepts in KB. Although the abduced ground hypotheses in Figure 3b are more clear than the dictionary produced by statistical induction in Figure 3c, the later is more general as it contains many sub-strokes which can compose more complicated written primitives.

Classification and Domain Adaptation

Datasets We use two typical classification datasets to conduct the experiments:

- **MNIST** (LeCun et al. 2001): This dataset consists of 28×28 binary images with 60,000 training and 10,000

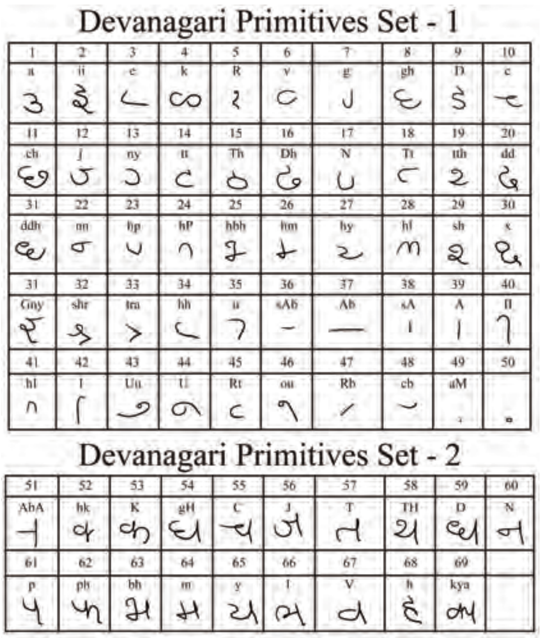
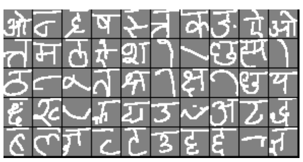
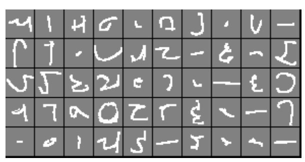


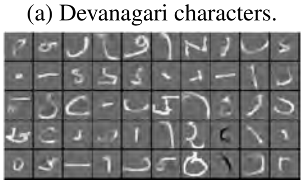
Figure 2: Human defined Devanagari primitives, reprinted from (Kopparapu and Lajish 2014).



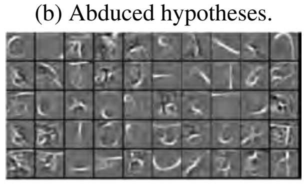
(a) Devanagari characters.



(b) Abduced hypotheses.



(c) Result of *LASIN*.



(d) Result of Sparse Coding.

Figure 3: Samples of results of Devanagari task.

test instances. In order to determine whether *LASIN* has the ability to learn from small data like human beings, we randomly sample 100 images for each class to create a sub-sampled training data.

- **Omniglot** (Lake, Salakhutdinov, and Tenenbaum 2015): Omniglot dataset consists of 105×105 binary images across 1628 classes with just 20 images per class. In order to make this domain transferable with MNIST, we rescale the Omniglot data to 28×28 images. The dictionaries are trained on small datasets Omniglot_small_1 and Omniglot_small_2 respectively. Different from the proposed model in (Lake, Salakhutdinov, and Tenenbaum 2015), we just use the raw images but not the pen trajectories.

Methodologies We adopt the routine of (Raina et al. 2007) to evaluate the learned dictionaries: they are used for coding the training and test data, then basic classifiers are trained and tested to evaluate the performance of each dictionary.

For *LASIN*, we adopt three kinds of knowledge base for logical abduction. The first one is the stroke definition in KB1, denoted as **stroke**. The second and third background knowledge bases **kmeans** and **spectral** basically talk about splitting characters into several parts by clustering all ink points for each image. They can be conveniently expressed by FOL knowledge base as well:

$$\begin{aligned} parts(C, S) \leftarrow\ & ink\_points(C, P), \\ & cluster(P, S, k). \end{aligned} \tag{3}$$

where P is the set of all ink points of a character C; $S = \{s_1, \ldots, s_k\}$ are the k ink-point clusters of C; $cluster = \{kmeans, spectral\}$ are the clustering approaches for **kmeans** and **spectral** respectively; $cluster(P, S, k)$ means S is obtained by clustering ink points P into k separate clusters. In the experiments we fixed $k = 2$, assuming that all characters should be split into 2 parts. The spectral clustering are compared because handwritten strokes can be regarded as 1-d manifolds embedded in a 2-d canvas.

The dictionary sizes are set at $|D| = 20, 50, 100, 200$, respectively. These sizes are not very large because we believe the effective dimension of handwritten characters should be small, involving some different strokes, their combinations and spacial relations. The hyper-parameters (turn limit and error threshold) of Algorithm 1 in the experiments are determined by cross-validation on training data.

The basic classifiers are multiclass SVMs with linear kernel implemented by libSVM (Chang and Lin 2011). We do not use complicated models because we try to keep the influence from classifier's power to be as small as possible, so that we can ensure all improvements are gained by introducing different kinds of human knowledge. All statistical models are trained with default parameter settings due to the same reason. The performance are evaluated with 5-fold cross-validation.

Tasks & Results The first task in this experiment is **classification**, which evaluates the quality of learned dictionaries by their performance on supervised classification tasks. The results are shown in the upper part of Table 1. On MNIST datasets, the performance of SC, which is consistent with (Yu and Ng 2010), is always worse than the proposed *LASIN* approaches. The classification accuracy on Omniglot datasets are quite low because they have more than 600 data-insufficient classes. On these datasets, *LASIN* with knowledge base **stroke** still performs best among all the compared approaches.

The second task in this experiment is **domain adaptation**. The dictionaries learned from source domains are used to code the data from target domains, then classification performance on target domains is evaluated. Here we report the results on MNIST and Omniglot_small_1 datasets (we omit Omniglot_small_2 because the results are quite similar). The results are shown in the bottom part of Table 1, where **M2O** denotes the adaptation from MNIST to Omniglot, and **O2M** denotes the inverse adaptation. From the results we can observe that *LASIN* with knowledge base **stroke** performs best

Size	$\lvert D\rvert = 20$				$\lvert D\rvert = 50$				$\lvert D\rvert = 100$				$\lvert D\rvert = 200$			
Method	SC	stroke	kmeans	spectral	SC	stroke	kmeans	spectral	SC	stroke	kmeans	spectral	SC	stroke	kmeans	spectral
MNIST	90.37	**91.66**	90.48	90.82	93.89	**94.88**	94.47	94.79	95.23	**96.27**	96.19	96.05	95.77	**97.01**	96.91	96.97
OS1	**16.46**	16.24	15.83	15.33	21.27	**21.82**	20.92	21.09	23.48	**24.74**	23.26	23.45	25.40	**26.64**	25.94	26.37
OS2	15.12	16.65	17.37	**17.43**	22.61	**22.90**	21.12	21.05	24.74	**25.07**	24.10	23.98	25.63	**26.29**	26.00	26.21
M2O	19.36	**20.69**	19.09	18.60	20.39	**23.38**	20.86	21.09	20.43	**23.19**	20.93	21.48	18.62	**24.63**	22.57	22.82
O2M	87.72	**88.14**	87.98	88.06	93.08	**93.69**	93.44	93.67	96.03	96.00	**96.11**	95.81	96.11	**96.39**	96.19	96.22

Table 1: Percentage of accuracy in classification (**MNIST**, **OS1** and **OS2**) and domain adaptation (**M2O** and **O2M**) tasks.

$\lvert D\rvert$	5×4	13×4	20×4	25×4	50×4	100×4
ACC	85.02	91.62	92.69	93.94	94.98	96.41

Table 2: Accuracy of patched Sparse Coding on MNIST Dataset.

in this task. The results of **M2O** show that if dictionary size is relatively small, the adapted dictionaries learned by *LASIN* are even better than unadapted ones in the **classification** task. A possible explanation is that the statistical induction on MNIST domain is more effective than the data insufficient Omniglot domain. An interesting conjecture is that, when dictionary size grows, the performance of strong-to-weak domain adaptations will decrease. This might be because when dictionary size grows, the extracted concepts from the strong domain become more and more ad-hoc in order to reduce the sparsity penalty. This conjecture worths a future investigation. Note that it also offers an evidence that model reuse can be very helpful rather than building a model from scratch in many situations (Zhou 2016).

Discussion

It is well acknowledged that human learning is more advanced than machine learning in several aspects. For example, owning to its ability to exploit an abundant supply of background knowledge, human can learn accurate models with a few training examples (Tenenbaum et al. 2011; Lake, Salakhutdinov, and Tenenbaum 2015). Since *LASIN* is proposed to exploit human knowledge, it is natural to ask whether can *LASIN* gain these benefits as human learning.

For the question on performance of the learned models, the indirect evaluation in previous experiments show that the representations learned by *LASIN* can boost the accuracy in supervised classification tasks. For the question on data requirement, we did some extra experiments on MNIST data, showing with Table 2. Because *LASIN* uses patch-based sparse coding, we also use patch-based SC for comparison. We sample $10,000$ 14×14 patches from all MNIST training images (which is far more than the training data used by *LASIN*) and used SC to learn patch dictionaries; then each training instance is split into 4 sub-images and each sub-image is coded with the learned dictionary for classification. If we compare them with *LASIN* according to the total length of the coded data, *LASIN* is always better than original patch-based SC. Even if we compare them according to patch dictionary sizes, *LASIN* is still comparable although its coded instances are just $1/4$ of patched SC in total code length. Hence background knowledge indeed can help reduce the requirement of data amount.

Another interesting question is how does the quality of background knowledge affect the learning results. From the Omniglot results in Table 1 we can observe that, although the performance of **stroke** is still better than SC, **kmeans** and **spectral** are sometimes worse than SC. This is because the assumption on the number of clusters to be 2 is acceptable for Arabic numbers but not appropriate for Omniglot data. Since the background knowledge of **stroke** is a better explanation of the data structure (although in a higher level), it is not surprising that it is superior to other approaches. Therefore, a wrong background knowledge can degenerate the performance of learning. A common example in human cognition is illusion, which is believed to be caused by the contradiction of our background knowledge with real situations (Solso, MacLin, and MacLin 2013).

Therefore, how to obtain the FOL background knowledge base is crucial to *LASIN*. Besides of using user-defined KB, like SRL and PILP, it is possible to use ILP techniques to learn FOL rules that bridging symbolic knowledge and raw data (Dai, Muggleton, and Zhou 2015). Ideally, the background knowledge base should be maintained by the AI system itself during its development, e.g., a sequence of easy-to-hard learning tasks where the starting primitives (like $ink/1$) are taught by humans.

6 Conclusion

To exploit human knowledge when learning from raw data, in this paper we formulate a novel task *concept extraction* aiming at using FOL background knowledge to constrain the representation learning in raw feature space. Inspired by human cognition, we propose the *LASIN* approach which combines logical abduction and statistic induction. Experimental results on handwritten character datasets validate its effectiveness. Experiments also suggest that by exploiting human knowledge, *LASIN* can learn good representations with smaller data. *LASIN* is a general-purpose approach with sufficient flexibility in implementation, e.g., the sparse coding ingredient can be replaced by other representation learning techniques such as deep learning. The choice of background knowledge base is important, and will be studied in the future.

References

Antanas, L.; van Otterlo, M.; Mogrovejo, J. O.; Tuytelaars, T.; and De Raedt, L. 2014. There are plenty of places like home: Using relational representations in hierarchies for distance-based image understanding. *Neurocomputing* 123:75–85.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE TPAMI* 35(8):1798–1828.

Bharath, A., and Madhvanath, S. 2010. *Online Handwriting Recognition for Indic Scripts*. London, UK: Springer. 209–234.

Chang, C.-C., and Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM TIST* 2:27:1–27:27.

Cireşan, D. C.; Meier, U.; Gambardella, L. M.; and Schmidhuber, J. 2010. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation* 22(12):3207–3220.

Curtin, R. R.; Cline, J. R.; Slagle, N. P.; March, W. B.; Ram, P.; Mehta, N. A.; and Gray, A. G. 2013. MLPACK: A scalable C++ machine learning library. *JMLR* 14:801–805.

Dai, W.-Z., and Zhou, Z.-H. 2015. Statistical unfolded logic learning. In *Proc. ACML*, 349–361.

Dai, W.-Z.; Muggleton, S. H.; and Zhou, Z.-H. 2015. Logical vision: Meta-interpretive learning for simple geometrical concepts. In *Late Breaking Papers of ILP*, 1–16.

De Maeyer, D.; Renkens, J.; Cloots, L.; De Raedt, L.; and Marchal, K. 2013. Phenetic: network-based interpretation of unstructured gene lists in e. coli. *Molecular BioSystems* 9(7):1594–1603.

De Raedt, L., and Kersting, K. 2008. *Probabilistic Inductive Logic Programming*. New York, NY: Springer.

Fiser, J.; Berkes, P.; Orbán, G.; and Lengyel, M. 2010. Statistically optimal perception and learning: from behavior to neural representations. *Trends in Cognitive Sciences* 14:119–130.

Getoor, L., and Taskar, B. 2007. *Introduction to Statistical Relational Learning*. Cambridge, MA: MIT press.

Hofstadter, D. R. 1985. *Metamagical Themas: Questing for the Essence of Mind and Pattern*. New York, NY: Basic Books, Inc.

Kakas, A. C.; Kowalski, R. A.; and Toni, F. 1992. Abductive logic programming. *Journal of Logic and Computation* 2(6):719–770.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA: MIT Press.

Kopparapu, S. K., and Lajish, V. L. 2014. A framework for online devanagari handwritten character recognition. *Preprint ArXiv* 1410.6909.

Lake, B. M.; Ullman, T. D.; Tenenbaum, J. B.; and Gershman, S. J. 2016. Building machines that learn and think like people. *Preprint ArXiv* 1604.00289.

Lake, B. M.; Salakhutdinov, R.; and Tenenbaum, J. B. 2015. Human-level concept learning through probabilistic program induction. *Science* 350(6266):1332–1338.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 2001. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, 306–351. IEEE Press.

Lee, H.; Battle, A.; Raina, R.; and Ng, A. Y. 2007. Efficient sparse coding algorithms. In *NIPS 19*. 801–808.

Magnani, L. 2009. *Abductive Cognition: The Epistemological and Eco-cognitive Dimensions of Hypothetical Reasoning*. Berlin: Springer.

Michie, D. 1988. Machine learning in the next five years. In *Proc. European Working Session on Learning*, 107–122.

Nitti, D.; De Laet, T.; and De Raedt, L. 2014. Relational object tracking and learning. In *Proc. ICRA*, 935–942.

Olshausen, B. A., and Field, D. J. 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381(6583):607–609.

Park, W. 2015. From visual abduction to abductive vision. In *Philosophy and Cognitive Science II: Western & Eastern Studies*. Springer. 141–153.

Peirce, C. S. 1955. *Philosophical Writings of Peirce*. New York, NY: Dover Publications.

Pyszczynski, T., and Greenberg, J. 1987. Toward an integration of cognitive and motivational perspectives on social inference: A biased hypothesis-testing model. In *Advances in Experimental Social Psychology*, volume 20. Academic Press. 297–340.

Raina, R.; Battle, A.; Lee, H.; Packer, B.; and Ng, A. Y. 2007. Self-taught learning: transfer learning from unlabeled data. In *Proc. ICML*, 759–766.

Russell, S. 2015. Unifying logic and probability. *Comm. of the ACM* 58(7):88–97.

Schacter, D. L.; Gilbert, D. T.; and Wegner, D. M. 2011. *Psychology*. New York, NY: Worth, 2nd edition.

Shivram, A.; Khot, T.; Natarajan, S.; and Govindaraju, V. 2014. Statistical relational learning for handwriting recognition. In *Proc. ILP*, 126–138.

Solso, R. L.; MacLin, M. K.; and MacLin, O. H. 2013. *Cognitive Psychology*. Pearson Education.

Tamaddoni-Nezhad, A.; Chaleil, R.; Kakas, A. C.; and Muggleton, S. H. 2006. Application of abductive ILP to learning metabolic network inhibition from temporal data. *Mach. Learn.* 64(1-3):209–230.

Tenenbaum, J. B.; Kemp, C.; Griffiths, T. L.; and Goodman, N. D. 2011. How to grow a mind: Statistics, structure, and abstraction. *science* 331(6022):1279–1285.

Tiercelin, C. 2005. Abduction and the semiotics of perception. *Semiotica* 2005(153):389–412.

Wang, W. Y.; Mazaitis, K.; and Cohen, W. W. 2013. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *Proc. CIKM*, 2129–2138.

Wielemaker, J.; Schrijvers, T.; Triska, M.; and Lager, T. 2012. SWI-Prolog. *Theory and Practice of Logic Programming* 12(1-2):67–96.

Yu, K., and Ng, A. Y. 2010. ECCV 2010 Tutorial: Feature learning for image classification.

Yu, K.; Zhang, T.; and Gong, Y. 2009. Nonlinear learning using local coordinate coding. In *NIPS 22*. 2223–2231.

Zhou, Z.-H. 2016. Learnware: on the future of machine learning. *Frontiers of Computer Science* 10(4):589–590.